Technical Report

CMU/SEI-93-TR-26
ESC-TR-93-200

Carnegie Mellon University
Software Engineering Institute

AD-A277 289
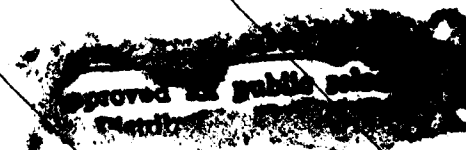
Case Studies of Software Process Improvement Methods

Daniel J. Paulish

December 1993

DTIC
ELECTE
MAR 22 1994
S   E   D

94-08935

94 3 21 021

Case Studies of Software Process Improvement Methods

Daniel J. Paulish

Resident Affiliate,
Siemens Corporate Research, Inc.

Accesion For	
NTIS CRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Dist. ibution /	
Availability Codes	

Dist	Avail and / or Special
A-1	

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

This technical report was prepared for the
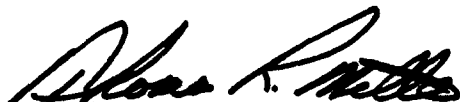
SEI Joint Program Office
ESC/ENS
Hanscon AFB, MA 01731-2116

The ideas and findings in this report should not be construed as an official
DoD position. It is published in the interest of scientific and technical
information exchange.

Review and Approval

This report has been reviewed and is approved for publication.

FOR THE COMMANDER

Thomas R. Miller, Lt Col, USAF
SEI Joint Program Office

Table of Contents

List of Figures

List of Tables

Preface

This report is an output of a joint Software Engineering Institute (SEI)/Siemens project in which Siemens software development organizations are being used as case study sites to measure and observe the impact of methods used to improve the software development process. The objective of the project is to quantify and better understand the more widely practiced methods used by industrial organizations to improve their software development processes.

The report is intended for software engineering managers and practitioners who are interested in improving their software development process. In this report, the author assumes that the reader has general knowledge of the structure and content of the SEI Capability Maturity Model for Software (CMM). If you are not familiar with the CMM, please refer to [Paulk 93].

The report describes the approach used for the case studies. It defines a number of basic measures to help organizations track the improvement in software development performance as the software development process is improved. The report contains the results and observations made for the Siemens software development organizations that were studied, and also contains a number of suggestions for improving the software development process based upon observation of the methods applied at the case study organizations.

Case Studies of Software Process Improvement Methods

Abstract: This report describes the case studies approach applied at a number of Siemens software development organizations to observe the impact of software process improvement methods. In addition, the report provides guidance to software development organizations that want to improve their processes. A set of organization performance measures are defined to help an organization observe its software process improvement over time. An approach is given for selecting software process improvement methods. The report concludes with a description of common implementation problems, and recommendations for organizations to improve their software processes.

1 Introduction

1.1 Project Background

In 1992 a joint project was initiated between the Software Engineering Institute (SEI) and Siemens to investigate the impact of software process improvement methods. There were two problem questions that motivated the project:

- How does one measure the result of software process improvement methods?

- How should an organization select methods for software process improvement?

A joint SEI/Siemens project on measuring software process improvement methods was initiated to integrate the methods developed at the SEI with actual practices used within Siemens software development organizations. This report is an output of this joint project. The project will identify specific process improvement methods that can be tailored to the current maturity level of the organization that wishes to improve. This project will also provide practical suggestions concerning the implementation and impact of process improvement methods in order to provide the foundation for continuous process improvement.

The goals of the SEI in this collaboration were:

- To obtain access to software engineering practices of industrial software development organizations in Siemens companies.

- To obtain methods for conducting case studies for SEI validation efforts.

- To gain access to Siemens process measurement research, standards, and practices.

The goals of Siemens Corporate Research in this collaboration were:

- To obtain access to SEI software process improvement methods and technology.

- To benefit from SEI's technology transfer mechanisms.

- To benefit from SEI's staff expertise and relationships as a technology center for software engineering process.

It is widely believed that an improved software development process results in higher quality products, which ultimately increases the ability of an industrial organization to compete in a competitive marketplace. The case studies described herein provide an opportunity to observe the practice and impact of methods for improving the software development process. It is the hypothesis of this project that the application of software process improvement methods will have a positive impact on the performance of a software development organization, as observed by relative results such as increased product quality and productivity, reduced schedule cycle times, improved morale, etc.

1.2 Scope of Report

This report describes a limited number of process improvement methods that have been commonly applied within industrial software development organizations, i.e., within the selected case study sites. All possible improvement methods cannot be adequately described herein; however, a subset of commonly applied methods has been described to give software engineering managers some background and guidance on methods that have been successfully applied in other organizations. The report summarizes lessons learned from organizations that have implemented these methods.

The software process improvement methods described have been selected from application within the Siemens case study sites. Because of the diversity of application domain, organization size, maturity level, location, etc., of the Siemens sites, it is believed that this report gives a reasonable description of current industrial practice. The case study approach has also been designed and described so that it could be applied to other industrial organizations.

It is anticipated that this report is the first of a series of reports covering Siemens experience implementing software process improvement methods. Future technical reports will report best practices from which others can learn. In addition, common barriers to overcome and observed methods for overcoming these barriers will be documented and summarized without identification of specific organizations.

1.3 Need for Process Improvement

The motivation to improve a software process usually results from a business need such as strong competition, increased profitability, or external regulation. Approaches to improve a software development process, such as those shown in Figure 1-1, are often initiated by an assessment of the current practices and maturity. A number of improvement methods are then recommended and implemented. The selection and successful implementation of the improvement methods are dependent on many variables such as the current process maturity, skills base, organization, and business issues such as cost, risk, implementation speed, etc. Measuring the impact and predicting the success of a specific improvement method are difficult. This is often due to environmental variables external to the method such as staff skills, acceptance, training effectiveness, and implementation efficiency. Once the improvement method is in place, there is also the question of what to do next. It is necessary to determine whether the method was implemented successfully, whether the process is mature enough to consider implementing additional methods, or whether the selected method is appropriate for use within the current process maturity level and environment.
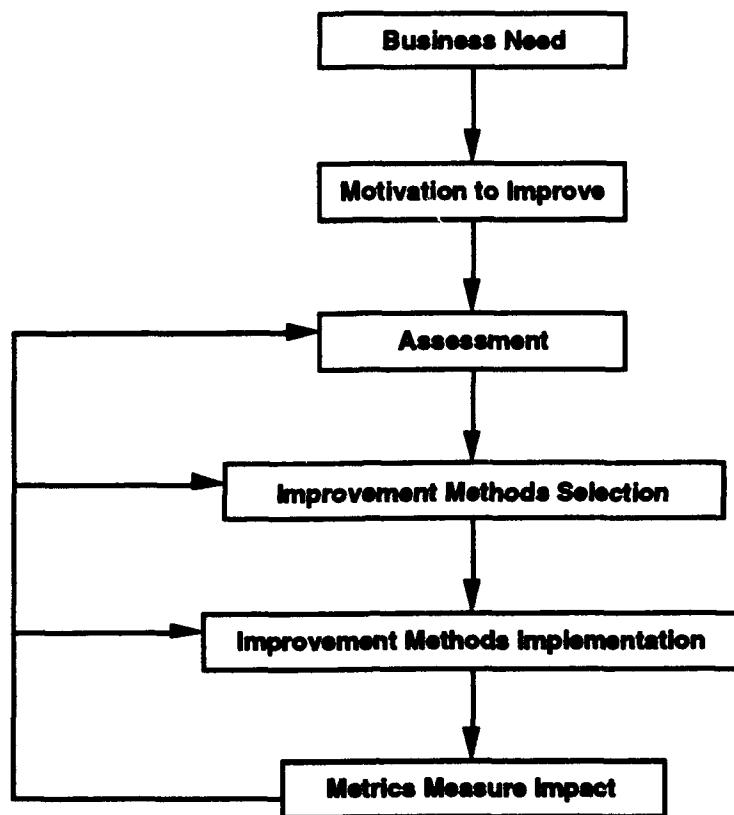
Figure 1-1. Process Improvement Approach

Many software engineering organizations today want to improve their software development process as a way of improving product quality and development team productivity, and reducing product development cycle time, thereby increasing business competitiveness and profitability. Although many organizations are motivated to improve, few know how best to improve their development process. There is a wide assortment of available methods such as TQM (total quality management), QFD (quality function deployment), FPA (function point analysis), DPP (defect prevention process), SWQA (software quality assurance), CM (configuration management), SRE (software reliability engineering), etc. This often creates confusion for software engineering managers with respect to which methods should be introduced at which points within their process evolution as shown in Figure 1-2.
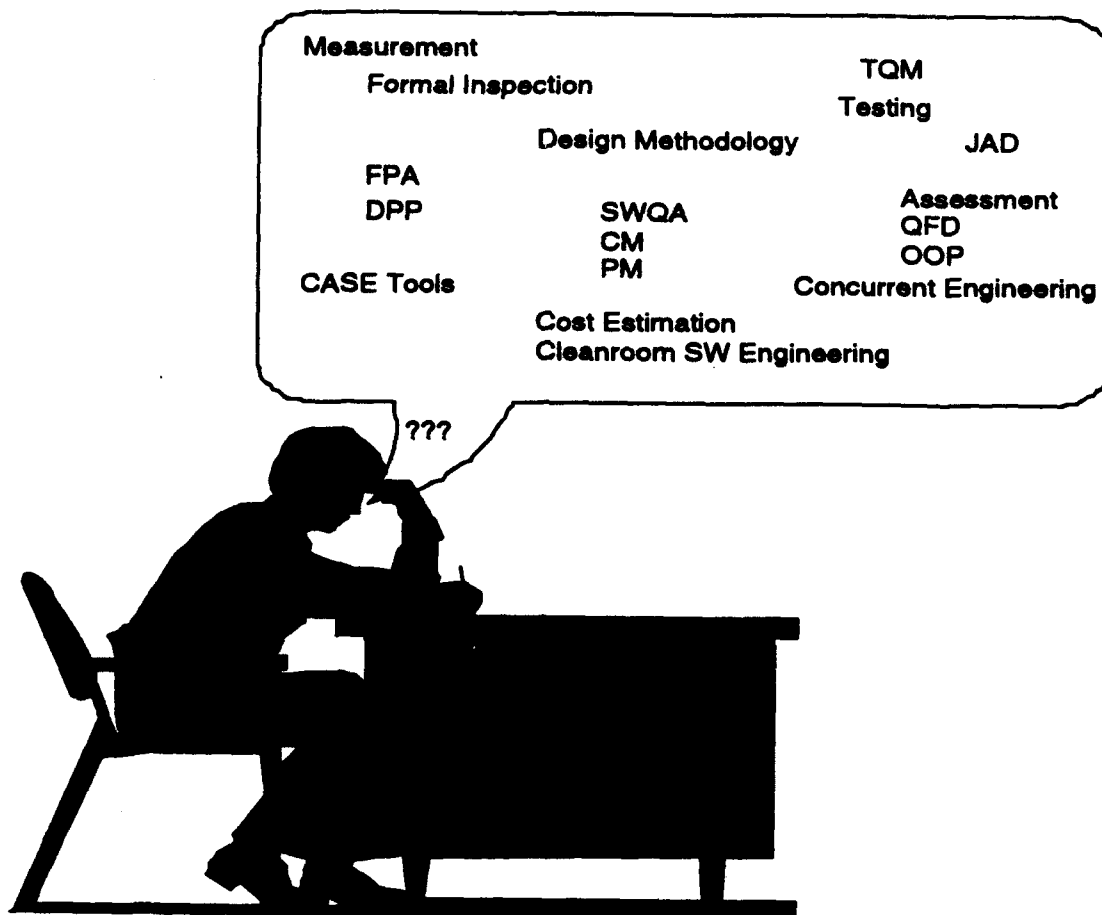
Figure 1-2. Which Software Process Improvement Methods?

1.4 What Is a Process Improvement Method?

A software process improvement method is defined as an integrated collection of procedures, tools, and training for the purpose of increasing product quality or development team productivity, or reducing development time. A software process improvement method can be used to support the implementation of a key process area (KPA) of the Capability Maturity Model (CMM) or to improve the effectiveness of key practices within a KPA.

Some example results of an improved software development process could include:

- Fewer product defects found by customers.

- Earlier identification and correction of defects.

- Fewer defects introduced during the development process.

- Faster time to market.

- Better predictability of project schedules and resources.

Software process improvement methods often require significant training and effort to introduce for application within a software development organization. Thus an organization must usually invest significant resources to introduce a process improvement method. In addition, there may often be barriers to adoption which an organization must overcome before one can observe a measurable impact resulting from the improved software process.

Some example software process improvement methods are summarized below.

- *Estimation:* This collection of methods uses models and tools to predict characteristics of a software project such as schedule and personnel needs before the project begins.

- *ISO 9000 certification:* ISO 9000 is a series of quality standards established by the International Standards Organization (ISO) for certifying that an organization's practices meet an acceptable level of quality control.

- *Software process assessment (SPA):* Assessment methods are a means of determining the strengths and weaknesses of an organization's software development process. Results include a "maturity rating," and findings of potential areas for improvement which are often implemented by a software engineering process group (SEPG).

- *Process definition:* These methods refer to the practice of formally specifying or modeling the software development process in a way that allows communication and analysis through its representation.

- *Formal inspection:* This method, pioneered by Michael Fagan at IBM in the 1970s, provides a technique to conduct review meetings to identify defects for subsequent correction within code or other documents.

- *Software measurement and metrics:* This collection of methods provides mechanisms for defining, tracking, and analyzing measures which can be used for controlling and improving the software development process.

- *Computer aided software engineering (CASE):* This collection of methods uses software tools for automating the software development process, particularly in the areas of design and analysis.

- *Interdisciplinary group methods (IGMs):* This collection of methods refers to various forms of planned interaction between people of diverse expertise and functional responsibilities who are working together as a team toward the completion of a software system. Example methods include nominal group technique (NGT), joint application design (JAD), groupware, group decision support systems (GDSSs), quality circles (QCs), and concurrent or simultaneous engineering.

- *Software reliability engineering (SRE):* SRE is a collection of methods using models for statistically predicting failure rates of a software system before it is released.

- *Quality function deployment (QFD):* This method is used to assist in defining software functional requirements that can best meet customer needs, distinguish resulting products from those of competitors, and consider implementation difficulty.

- *Total quality management (TQM):* This collection of methods is oriented towards improving the quality culture of the organization, including assistance to define, improve, and track goals.

- *Defect prevention process (DPP):* This method, pioneered at IBM in the 1980s, assists in categorizing defects so they can be systematically removed and avoided in future software development products and activities.

- *Cleanroom software development:* Cleanroom is a software production method which originated in the Federal Systems Division of IBM in the late 1970s and early 1980s. Cleanroom combines practices of formal specification, nonexecution-based program development, incremental development, and independent statistical testing.

2   Case Studies Approach

2.1   Introduction

This section discusses the approach taken to select, interview, and collect data from the Siemens case study sites. It also addresses some of the issues and concerns associated with participation in this project, such as information protection, required investment, and benefits to the Siemens development organizations.

Figure 2-1 summarizes the approach used for the case studies. A project overview and initial interview were conducted at the sites in November-December, 1992. The measurement baseline was established for some of the sites during March-April, 1993. Follow-up interviews were conducted at the sites during July-August, 1993. The sites will continue to be measured and observed for the next few years with measurement data collected and interviews conducted at least annually.

```
        ┌─────────────────────┐
        │  Project Overview & │
        │  Initial  Interview │
        └─────────────────────┘
                   │
                   ▼
        ┌─────────────────────┐
        │ Measurement Baseline│
        └─────────────────────┘
                   │
        ┌──────────┤
        │          ▼
        │  ┌─────────────────────┐      ┌─────────────────────┐
        │  │ Follow-Up  Interviews│─────▶│    Conclusions &    │
        │  │   & Measurement     │      │    Observations     │
        │  └─────────────────────┘      └─────────────────────┘
        │          │
        └──────────┘
```
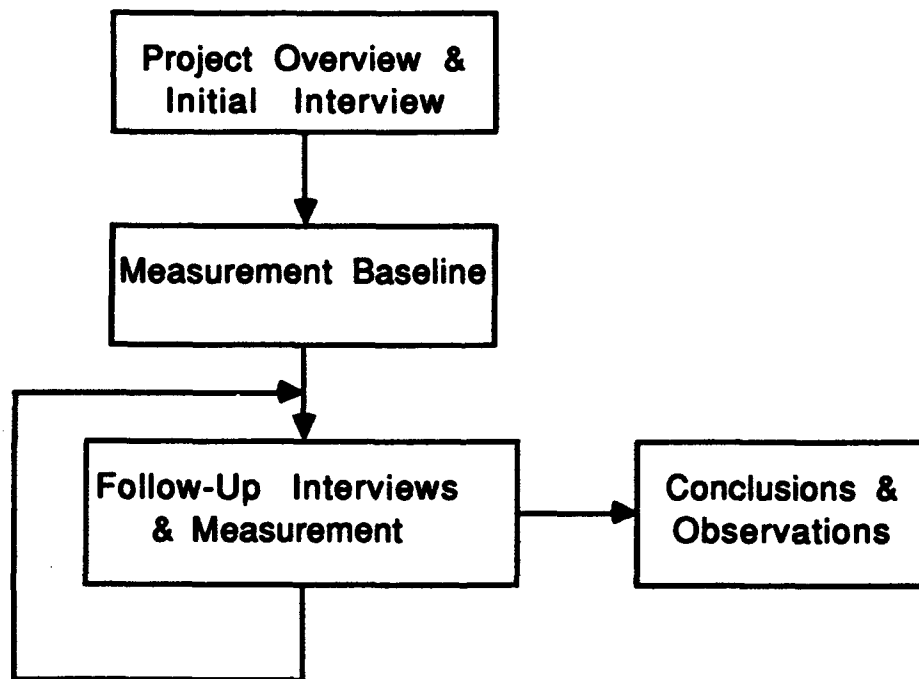
Figure 2-1. Case Studies Approach

2.2 Site Selection

A number of Siemens software development organizations were selected as case study sites to investigate the impact of selected process improvement methods. A limited number of basic measurements were made to capture the current performance of the development organization with respect to development team productivity, process maturity, and product quality. The selection criteria for Siemens case study site locations are summarized below:

- Siemens software development organizations were selected to obtain a large variance of application domains, size, product complexity, etc., for the case study.

- Organizations were selected in multiple countries, specifically the USA and Germany.

- Organizations were selected in which the author had personal contacts, either established from prior projects, or through contacts made through the corporate software process improvement competence center.

- Organizations were selected that were relatively dedicated to and had an interest in software process improvement.

A number of process improvement methods were selected by the case study site organizations for implementation within the software development organization. The methods were chosen based on the current maturity level, skills base, organization structure, and business issues for each organization. Each development organization has been visited twice, and they will be revisited approximately annually, at which time the basic measurements will be recalculated. This will provide some quantitative data concerning the impact of the selected process improvement methods. In addition, lessons learned from the implementation and impact of the process improvement methods have been and will be captured and documented. In particular, observations concerning the use of the methods have been and will be captured including soft factors such as the impact on staff morale, quality culture, motivation, etc.

2.3 Interview and Data Collection Approach

The data for the case study investigation are collected from site interviews and documentation supplied by the case study site. Each initial site visit consisted of an overview of the project (approximately 1 hour), and then an interview which required from 1-2 hours depending on the amount of discussion. The questions used to guide the initial interviews are given below.

Practices/Process/Environment:

1. What types of products are developed here?
2. How many people here are involved with software product development?
3. What size are the developed products?
4. How long does it take to develop a new release?
5. Can you describe your current company quality culture?

Improvement Methods:

6. What methods have you used to improve your software process?
7. What methods are you planning to use to improve your software process?
8. How will you introduce the selected software process improvement methods?
9. What problems are you anticipating in improving your process?
10. How much will you invest in software process improvement?

Performance Measurements:

11. Has an assessment been made of your software process?
12. What is your current maturity level?
13. What is your current product quality?
14. What is your current productivity?

A few months after the initial interview, each site was asked to perform a measurement baseline in order to determine the current performance of the organization. The data requested are described in Chapter 4 of this report. These data were viewed as confidential by all the organizations, and in some cases the data were considered to be too sensitive for discussion external to the organization. No attempt was made to compare or contrast data across the organizations for the purpose of comparing organizational performance. The degree of difficulty that each organization experienced in generating the performance data varied considerably depending on the maturity level and degree of application of quantitative measurement approaches used within their development process.

Follow-up interviews have been and will be conducted with the organizations, and measurement data will be collected and updated in order to observe the impact of process improvement over time. We anticipate that the organizations will need to be observed over a period of four to five years in order to see an improvement in performance data. This is a consequence of the relatively long product development cycle times. Performance is normally calculated for a project or product release, such that the trend of data can only be observed by successive releases over time. Interviews will be conducted on roughly an annual interval depending on availability of staff and frequency of interaction.

The data collected from a case study site will often be for a recent project or projects for which some field detect rate data have been collected for one year. Thus, the term *site* does not imply a strict physical location. In some locations, multiple projects may be reported, while some projects may be managed over multiple physical sites often within multiple countries.

2.4 Information Protection

Information discussed and reported in the case studies is being protected in accordance with the desires of the case study organization. The most desirable approach is for an open discussion of lessons learned and results using the existing Siemens procedures for review and release of external data. However, some organizations may wish to withhold actual measurement data by normalizing the results. For example, an organization could report that their field defect density was X defects/KLOC before implementing formal inspections, and after one year of implementing formal inspections the field density decreased to 0.3X defects/KLOC. In addition, some organizations may wish to withhold their company identity. In some cases, organizations are willing to discuss the lessons learned with introducing specific process improvement methods, but they view their performance data as confidential and not to be discussed outside their organization.

2.5 Required Investment

The case studies are designed to be as noninvasive as possible respecting the limited staff time available within any product development organization. Each case study organization requires a point-of-contact. Interviews are conducted with the point-of-contact and any other key staff members necessary to gather the measurement data and describe current process, practices, improvement methods selected, and lessons learned. In addition, the case study organization must review the case study description reports for external release.

It is also suggested by the author, but not required for this project, that the case study organization consider undergoing an SEI software process assessment (SPA) [Humphrey 89] as a proven technique for identifying software process improvement activities. The investment required to conduct an SEI assessment can be substantial depending on the size of the organization and the approach selected. In general, a one-week on-site evaluation is necessary using a dedicated assessment team which would conduct interviews with a representative set of project leaders and practitioners. The output of the assessment would include not only a determination of the current process maturity, but more importantly, a set of recommendations for process improvement.

2.6 Benefits to Siemens Development Organizations

The benefits to a Siemens development organization of participating as a case study site for the SEI/Siemens joint project are summarized below.

- *Transferring SEI technology:* The project will provide the Siemens development organization with better access to SEI technology. The SEI's mission is to provide leadership in advancing the state of the practice of software engineering to improve the quality of systems that depend on software.

- *Generating positive publicity:* It is intended that the case studies material will be published and promoted for other organizations to learn from experience. Publication of the case studies will enhance the reputation of the Siemens organizations described.

- *Consulting:* Informal consulting on process improvement will be provided to the case studies organizations. This consulting could range from answering questions to training, especially in the area of software measurement.

- *Supporting process improvement and sharing best practices:* The Siemens case study organizations will have the opportunity to learn what is being done in other parts of Siemens in the U.S. and Germany to improve the software development process.

3   Case Studies Description

Ten potential Siemens case study sites have been identified, visited, and interviewed. The sites exhibit a high degree of diversity with respect to product application, staff size, maturity level, etc. The sites are located in both the U.S. and Germany, which provides the opportunity to observe the effect of cultural variables upon the software development organization. A two-tier approach has been taken with the sites concerning the degree of interaction and observation. The majority of organizations will require minimal interaction primarily through the scheduled interviews and occasional telephone follow-up. Two organizations have been identified as candidates for extensive observation. The Electromedical Group (SME) of Siemens Medical Systems in Danvers, Mass. is planning a software quality initiative which will result in the development of a number of planned actions for software process improvement. Siemens Private Communication Systems (PN) is planning a new product release which will involve multinational teams residing in Munich and at two locations in the U.S. Project management has expressed an interest in working closely with the SEI project since the product release planning and implementation will also address process improvement methods. In addition, a corporate thrust has been initiated to investigate methods for improving overall product development efficiency.

The diversity of the case study sites can be observed from the raw data summarized in Table 1, the site characteristics in Table 2, and the software process improvement methods used in Table 3. Note that no conclusions can be made from these tables since multiple products, projects, and releases are usually being worked on simultaneously by the software staff. Brief summaries of the selected software development organizations are given below.

3.1   Siemens Private Communication Systems (PN)

Siemens Private Communication Systems designs, develops, and sells communication systems throughout the world. These systems are known as telephone switching systems, private telephone exchanges, and private branch exchanges (PBX). The Siemens products are called HICOM™ systems. This Siemens business is made up of approximately 26,000 employees. Nearly 700,000 systems have been installed for 400,000 customers worldwide.

Approximately 800 employees in Europe are involved with software development. New software versions, depending on the application, typically take 1-2 years to develop. The product sizes are approximately 1-4 MLOC, and represent an investment of 100-300 staff-years to develop. Most of the code is written in CHILL, and a large part of it is shared among the various versions within a product line.

In addition to Europe, distribution and development activities in the United States are substantial having resulted from the acquisition of Rolm from IBM. Major software development centers are located in Munich, Germany; Santa Clara, California; and Boca Raton, Florida.

PN has invested in the software process methods of formal inspection, measurement, and project management training. It is particularly interested in implementing multi-national, twenty-four hour product development exploiting the existence of its multiple-site development centers located in different time zones—particularly Germany, Florida, and California. Towards this goal PN has already implemented its development computing facilities so that they can be shared among multiple development locations.

Table 1: Site Raw Data

Organization	Location	Software Staff Size	Product Size (LOC)	Schedule (Mos.)
ÖN ÖV	Munich	1600	5M	27
PN	Munich	800	4M	30
AUT	Erlangen	250	300-600K	8-12
SNI	Munich	1200	3M	24-26
SSC	Boca Raton, FL	500	3.7M	25-27
SIA	Johnson City, TN	45	600K	14-24
SME	Danvers, MA	75	150K	18
SGI-PACS	Hoffman Estates, IL	65	332K	6-8
ANG	Hoffman Estates, IL	25	234K	18-19
NUC	Hoffman Estates, IL	30	423K	30-36

Table 2: Case Study Site Characteristics

Characteristic	Range
Staff size	25-1600
Product size	150 KLOC-5 MLOC
Schedule cycle time	6-36 months
Maturity level	1-4
Software process improvement methods used	ISO 9000-DPP Most widely used methods: formal inspection, measurement, ISO 9000

Table 3: Software Process Improvement Methods

Org.	Location	Software Process Improvement Methods Used
ÖN ÖV	Munich	Inspection, measurement, process training, assessment
PN	Munich	Inspection, measurement, multinational development
AUT	Erlangen	Inspection, measurement
SNI	Munich	Inspection, CASE, DPP
SSC	Boca Raton, FL	Inspection, measurement, TQM, ISO 9000
SIA	Johnson City, TN	SEPG, CASE, measurement, ISO 9000
SME	Danvers, MA	Inspection, measurement, process definition
SGI-PACS	Hoffman Estates, IL	Process definition, assessment, ISO 9000
ANG	Hoffman Estates, IL	QFD, assessment, ISO 9000
NUC	Hoffman Estates, IL	Process definition, assessment, ISO 9000

3.2 Siemens Automation (AUT)

Siemens Automation develops factory automation systems products for applications such as computer integrated manufacturing (CIM). Approximately 20,000 people are employed worldwide by this Siemens business. The business headquarters is located in Nürnberg, Germany, with larger facilities also located in France, Austria, India, and the United States.

The specific site selected for the case study is located in Erlangen, Germany. The types of products that are developed are automation systems for machine tools and robots. The software development activities can be described as the development of embedded real-time systems. The software is typically written in C, C++, and Assembler on PC-based workstations, and then embedded as PROM-based software within microprocessor-controlled products.

This organization initiated the use of metrics using the PYRAMID seven-step metrics introduction program approach in 1991 [Möller 93]. After application to the pilot project organization (for approximately six months), the metrics program was revised based upon the pilot experience, and then implemented throughout this organization. The organization has also trained its staff on the use of formal inspections.

3.3 Siemens Nixdorf Informationssysteme (SNI)

This site, located in Munich, is responsible for systems software development within Siemens Nixdorf Informationssysteme (SNI). SNI was formed in October, 1990 by the merger of the Data and Information Systems Group of Siemens AG and Nixdorf AG. This organization has possibly one of the most mature software development processes within Siemens. For example, it is believed that this group's application of software measurement may be one of the earliest applications of this software process improvement method found in Europe.

Systems software refers to software that is developed for controlling the utilization of general-purpose computer systems. It includes software such as operating systems, computer language compilers, database management software, utilities, and data communications software. The primary operating systems developed and maintained are BS2000™, which is an operating system that runs on Siemens mainframes, and SINIX™, which is a Siemens supported version of the UNIX™ operating system.

The software products developed range in size from approximately 10,000 lines of code for small utility applications to more than 3 million lines of code for a complex operating system. Computer languages that are primarily used are C, SPL, and Assembler.

There are approximately 3,000 software engineers and support specialists involved in developing and maintaining these system software products. Many of the developers are located in the Munich and Paderborn areas in Germany, although there are development centers located around the world. Some of the larger satellite development centers in Europe are located in Namur, Belgium and Vienna, Austria. The case study will focus on the activities of the approximately 1,200 software developers in Munich. This group has recently completed a pilot project on defect prevention process (DPP), which when generally implemented, could meet some of the characteristics of a CMM level 5 organization.

SNI has also substantially invested in computer aided software engineering (CASE) tools and training over the past few years. It has implemented an extensive training program on formal inspections. This organization has observed a reduction in half of field defect rates within the last three years, which is believed to have resulted primarily from the application of formal inspections.

3.4 Siemens Stromberg-Carlson (SSC)

Siemens Stromberg-Carlson provides telephone operating companies with advanced, high-quality public telecommunications networks. The company designs, develops and manufactures central office digital switching systems, cell switching systems, packet switching systems and transmission equipment. The case study site will observe the practices of the EWSD™ product development group located in Boca Raton, Florida consisting of about 500 staff members involved with software development. The case study site does not include the development activities located in Lake Mary, Florida. Siemens Stromberg-Carlson develops its switches for the U.S. market reusing much of the base code developed at ÖN ÖV Munich. Features required for the U.S. market are then added, integrated, tested, and maintained.

SSC has recently implemented a metrics program, and they have trained all staff members on Crosby quality methods and formal inspections. The metrics program includes periodic publishing of a metrics newsletter and on-line access to the measurement data. This organization has reached a process maturity level such that it is now considering implementing a pilot defect prevention process (DPP) project. SSC is also planning for ISO 9000 certification. There is substantial management support for process improvement, and SSC staff have been attending training on a number of software process improvement methods which are also being considered for implementation.

3.5 Siemens Industrial Automation (SIA)

Siemens Industrial Automation develops programmable controllers for factory automation systems. The company, consisting of about 1,100 employees, was formed in late 1991 as a result of Siemens acquisition of the Industrial Controls Division of Texas Instruments in Johnson City, Tenn. The case study involves only one engineering department consisting of about 45 developers. These developers are developing relatively sophisticated products involved with a CASE tool for process control application programming (APT™), and supervisory control and data acquisition monitoring systems (TISTAR™).

The specific development department being used for the case study has a relatively sophisticated small team development process. The department manager and point-of-contact for the case study is a member of the SIA Management Steering Committee on Process Improvement. SIA has initiated activities for process improvement such as forming a software engineering process group (SEPG), using CASE tools, training staff in formal inspection, and implementing measurement. SIA has recently obtained ISO 9000 certification.

3.6 Electromedical Group (SME) of Siemens Medical Systems

The Electromedical Group (SME) of Siemens Medical Systems designs, develops, and manufactures patient-monitoring systems for use in hospitals throughout the world. The patient-monitoring products are typically used in intensive care units, cardiac care units, and operating rooms within hospitals.

Development teams are currently developing patient-monitoring products depending on the application of the device in the hospital (e.g., bedside monitors, central nurses' station, recorders, and displays). The devices utilize microprocessor-based hardware platforms using Motorola and Intel processors, and they utilize real-time operating systems. The devices are connected within the hospital for real-time data collection and distribution using local area networks.

The code is written primarily in C with some assembly language code for time-critical applications. The software development environments used include Sun Microsystems workstations, personal computers, and VAX™ minicomputers running SunOS™, MSDOS™, and VMS™ respectively.

High quality code is required to provide customer satisfaction and to ensure patient safety. To increase software quality, the organization has emphasized development methodology with carefully controlled specifications and reviews, better testing procedures and tools, and the use of quality metrics to measure progress. The products developed and production process techniques utilized are regulated by the U.S. Food and Drug Administration.

This organization is currently focused on process definition, formal inspection training, estimation, and measurement.

3.7 Siemens Gammasonics (SGI) - PACS

This SGI Division is involved with developing picture archiving and communications systems (PACS) for hospitals. Siemens PACS is a comprehensive digital information system integrating a wide spectrum of digital and analog modalities. These systems are used in order to streamline radiology operations within a hospital by introducing electronic image acquisition, transmission, storage, retrieval, and display capabilities.

This software organization has undergone a very rapid buildup of staff over the past few years with another twelve members to be added this fiscal year. This has resulted in a small project process which may be difficult to scale up to a larger organization. This coupled with the fact that the product is new and not yet in a maintenance mode, indicates an organization in transition. Their plan is to better document their current development process, and then use this material for training new and existing staff on

the characteristics and requirements of their process. An SEI style software process assessment was conducted in June, 1993. PACS is also planning for ISO 9000 certification.

3.8 Angiography (ANG) Group of Siemens Medical Systems

This group is involved with developing products for blood vessel imaging. The products are sold under the name Hicor™. Using low invasion techniques, these systems provide physicians with information necessary to diagnose and to plan therapies that circumvent surgery. They provide real-time images for monitoring of medical diagnostic and therapeutic angiography procedures such as angioplasty.

This organization exhibits a stable environment with staff that has in many cases worked together for a long time. The average staff age is greater than the other divisions in SGI, and there is a high degree of confidence concerning both process and performance. The business environment is considered stable and successful.

It was suggested that the organization could achieve higher levels of the CMM through more effective utilization of their measurement data for in-process measurement and control. An SEI style software process assessment was conducted during August, 1993. ANG applied quality function deployment (QFD) during the definition of system functional requirements for its most recent major product development. ANG is also planning for ISO 9000 certification.

3.9 Nuclear Medicine (NUC) Group of Siemens Medical Systems

This group develops nuclear medicine systems consisting of gamma cameras and computers. These systems provide whole body and organ specific imaging capability. This organization exhibits many of the business pressures observed at other case study sites in that there is extreme schedule pressure to deliver additional functionality to customers. Thus, the need for new features along with defect corrections has resulted in very frequent intermediate releases. This, in turn, often results in staff pressure and customer-driven reactive development.

This group appears to have a great motivation for process improvement. This group has defined a development process which is currently being introduced and debugged. This process definition work will continue, and basic measures should be added to support the process. An SEI style software process assessment will be conducted during 1994. This organization is also planning for ISO 9000 certification.

4 Performance Measures for Software Development Organizations

4.1 Introduction

In order to observe the impact of software process improvement over time, it was necessary to identify a basic set of performance measures for software development organizations. By observing the trend of such performance data, one may conclude whether the organization's performance is improving as a result of the methods implemented for improving the software development process.

Measures should be defined within an organization in accordance with the organization's goals [AMI 92], [Möller 93]. For example, an organization whose goal is to reduce development cycle times should have a highly visible measure of project development schedule time. Thus, the organization performance measures defined below are not intended to be "good" measures for every organization. Rather they are a basic set of more commonly applied measures that should assist in data collection and observation of the case study sites. It is not necessary for all sites to measure in the same way. However, each site must measure consistently over time.

The performance measures defined in this chapter were provided to the case study sites with a request to provide data as per the definitions. It was suggested that the organizations provide such data to the study, although data using differently defined measures would also be welcome. What we desire for the study is consistent data reporting over time, such that an organization's performance improvement can be observed as a result of software process improvement. This assumes that the products developed by the organizations will be of similar complexity over the observation time period (3-5 years). This is probably a reasonable assumption since the sites selected are all involved with maintaining mature products as well as developing newer versions of these products.

Most of the sites that supplied data utilized the definitions provided, or they had already been tracking similar data resulting from previously shared software measurement training across Siemens business units. It is suspected that the accuracy of the reported data is better for the organizations that have established measurement programs as compared to organizations that collected the data for the first time for this study.

4.2 Data Collection and Analysis

Organization performance is very much dependent on the application domain of the developed products. Different performance can be expected for organizations de-

pending on the complexity and application characteristics of the products being produced, the activities being performed by the development organization, the development environment employed, and the business situation. For example, different performance in productivity should be expected for different types of development such as new product development, enhancement, migration, conversion, or maintenance. Business factors (such as whether or not the business is profitable) and external regulatory requirements also affect the performance of an organization. It is not the intent of this project to establish what is good performance or to compare performance of organizations, but to be able to observe changes in performance over time within a specific organization.

In this report, a software development organization is defined as a collection of people who produce software. Thus, a software development organization could be a software engineering department or project team. One would anticipate that a software development organization's performance would improve as their software development process was improved. This could be observed by comparing performance measures calculated for current development as compared with prior developments, provided they are doing work of similar difficulty. These developments are usually called releases, projects, or products within the development organization. We will collect performance measures on successive developments for producing a software entity which is delivered to users (customers).

In organizations where many independent projects are simultaneously ongoing, the project performance measures could be averaged for the overall organization and then compared at time intervals (e.g., for each fiscal year). The performance measures will need to be observed for a multiyear time period, since process improvement methods often require substantial time for training and acceptance within the organization, and development cycle times may be long.

4.3 Primary Data

It is suggested that certain primary data be collected for each software development organization as summarized in Table 4. These data are similar to the recommended "SEI core measures" [Carleton 92]. The baseline data should initially be collected for the last major product software release that has been used by customers in the field for at least one year.

Table 4: Primary Data

Measure	Units of Measurement
Defects	Number of defects found per phase
Product size	Lines of code (LOC)
Effort	Staff-hours
Schedule duration time, actual	Months per phase
Schedule duration time, estimated	Months per phase

The number of *defects* found per phase and in total should be counted. Each organization will have unique defect counting rules, describing what is counted as a defect and during which activities they are counted. For guidance, [Florac 92] should be considered, in which a software *defect* is defined to be any flaw or imperfection in a software work product or software process. When found in executable code, a defect is frequently referred to as a *fault* or *bug*.

The definition of a phase will vary depending on the organization's software development process. For simplicity, we will count faults during three generalized phases.

- The implementation phase will begin when an approved functional specification is available and high-level design (HLD) has begun. The implementation phase would normally include process steps such as high-level design, detail design, coding, unit test, and integration.

- The qualification testing phase will begin when an integrated software system is available for testing, usually performed by an independent test function.

- The field use phase will begin when the first software system is delivered to a customer and faults will be counted for one year of customer use after the first installation is made.

These generalized phases are shown in Figure 4-1.

No. Defects Found

Implementation

Field Use

1 Year

Qualification
Testing

Begin
HLD

Test
System

Install

Time
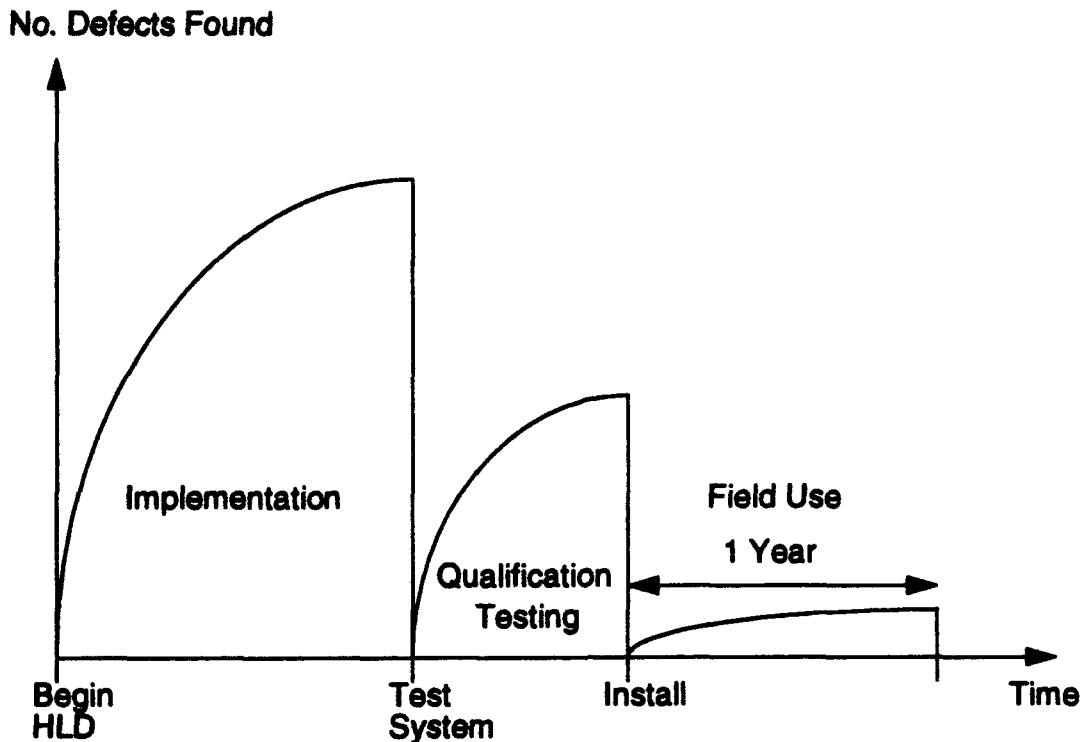
Figure 4-1. Generalized Process Phases

Product size should be measured using function points [IFPUG 90] or lines of code. For lines of code (LOC) measurement definition, the Siemens LOC Norm [Siemens 89] or [Park 92] should be considered. The measurement should be made at the time of product release to customers. The desired measure to be reported is the count of lines of code excluding comment and blank lines.

Effort should be counted in accordance with the existing organization accounting and time reporting procedures. Effort should include the work performed by all staff involved with software development including test, quality assurance, and documentation specialists. Effort should be counted for the duration of the phases defined above; i.e., effort involved with the product development from the beginning of high-level design to one year after first customer delivery. Guidance on effort counting can be found in [Goethert 92].

Schedule duration time for the implementation and qualification testing phases should be reported. We will define the schedule cycle time as the calendar time between the start of high-level design to first customer delivery or the sum of the schedule duration times for the implementation and qualification testing phases. In addition to the actual time required for the last release, it would be desirable to report the times that were

estimated for these phases prior to implementation, i.e., the estimated schedule. Guidance on schedule information reporting can be found in [Goethert 92].

4.4 Environmental Data

Improving the software development process will improve the quality of the software products resulting from that process. It is important to realize, however, that process is only one of the factors that can be controlled for improving software quality. This can be seen in Figure 4-2 developed by [Kellner 92]. Factors other than process that influence software quality include the skills and experience of the people developing the software, the technology used such as CASE tools, the product complexity, and environmental factors such as schedule pressure, communications, etc.
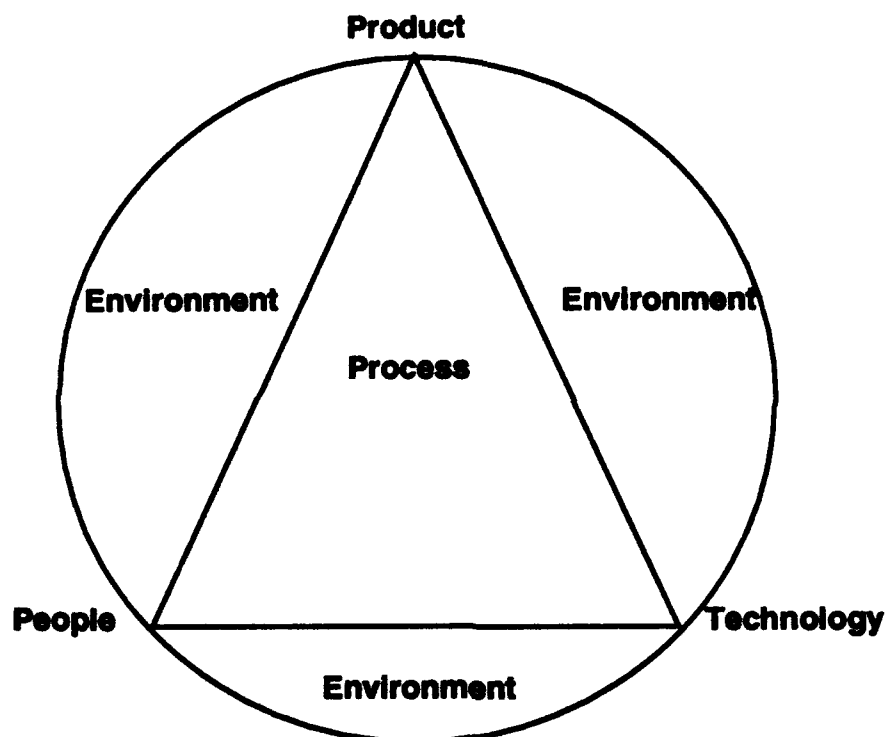
Figure 4-2. Software Quality Determinants

It is desirable to measure some of the characteristics of the organization environment in which the software was developed. This data will be used to attempt to identify key influences on the development organization's performance, and to help understand the environmental context in which the products are developed. The data collected will be somewhat "soft." Within the discipline of software measurement, environmental measures are somewhat more poorly defined and less commonly applied than product and process measures. It is anticipated that these measures may change as more experience is obtained.

The basic set of environmental measures is summarized below in Table 5.

Table 5: Environmental Data

Measure	Units of Measurement
Staff size	Number of people
Staff turnover	% people who left in last 12 months
Maturity level	Estimate or from assessment: 1-5
Morale	Subjective response class: poor, average, good, excellent

Organization *staff size* should be measured at the time of the baseline measurements or at a recent convenient time point (e.g., last month, quarter). Staff members should be counted in the case study software development organization who perform the work included in the effort measure above. For example, staff members such as software engineers, testers, quality specialists, and documentation specialists should be counted. The count should include full-time, permanent employees, not temporary contractors or part-time employees. It is well accepted that larger organizations are generally more difficult to manage and control, and that software development processes for small organizations can be much different than for large organizations.

Staff turnover should be measured as an indicator of the stability of the organization. Staff turnover will be defined as shown in the following equation:

Staff turnover = % (No. staff members who have left in the last 12 months)/Staff size.

A high degree of staff turnover could be the result of downsizing activities or extreme business conditions that result in organization stress. This measure can also give some insight into how long staff members remain within the organization. If many new hires replace the staff members who left, an organization may have training and staff assimilation challenges. Some organizations may also wish to calculate the staff turnover independently for management and technical staff.

The software process *maturity level* of the organization should be determined in accordance with the Capability Maturity Model (CMM) [Paulk 93]. We have suggested that the case study site organizations conduct self assessments around the time of the baseline measurements, and then biannually thereafter. If an assessment has been done of the organization within the last two years, the assessment level should be reported. If an assessment has not been done, then a current estimate of process maturity level should be made. These data are desirable for correlating the process improvement methods selected with the process maturity of the organization.

It would be desirable to measure the staff *morale* of the organization to see if this has a significant influence on organization performance. This is not easy to measure, although some organizations conduct periodic morale and opinion surveys. If such a survey is used, then the data would be readily available. If not, an estimate should be made of the current morale. Since this is a subjective metric, we suggest that a four level response class scheme be used - poor, average, good, and excellent.

4.5 Performance Measures

A basic set of performance measures for a software development organization can be calculated from the primary data as summarized in Table 6. The measures will be calculated for each successive project, release, or product and compared with the measures of the prior project, release, or product over time. It is not the intent of this investigation to compare measures across organizations, but to observe the performance improvement within an organization. Each of the measures is described below.

Table 6: Organization Performance Measures

Measure	Units of Measurement	Primary Data Used
Defect detection distribution	% Defects found per phase	Defects
Defect rate	Defects/KLOC per phase	Defects, product size
Project productivity	LOC/Staff-hour	Product size, effort
Schedule cycle time	Months	Schedule duration time, actual
Schedule adherence	%(Estimated-Actual)/ Estimated	Actual and estimated schedule duration times

The first two organization performance measures are calculated from the defects primary data. *Defect detection distribution* is calculated as the percentage of faults found in each development phase. *Defect rate* is calculated as the number of defects for each development phase divided by the product size in thousands of lines of code (KLOC). The defect rates should decrease for later phases of the product development process. If the number of users of the product is large, one can assume that almost all of the product defects will be found during its lifetime, and that many of the faults will be found during the first year of use. The defect rate for the field use phase is thus a reliable measure of product quality [Jilek 92]. Exceptions to this would be the cases when a product's quality is very bad, and the users quickly stop using the prod-

uct and stop reporting defects, or the number of users is small. The *defect insertion rate* is defined as the sum of the defect rates over all phases.

Project productivity is a basic organization performance measure. A common method of calculating project productivity is to divide the product size by the effort. Organizations who develop multiple releases for the same product may wish to count delta lines of code for their release productivity calculation. Delta lines of code usually count changed and added lines of code for a release as compared with the prior release. Productivity could also be calculated using function points rather than lines of code. In some cases project productivity calculated using the total product size is called product productivity. If delta lines of code are counted, the project productivity is often called process productivity. One must also define rules for handling reused code. Previously tested code integrated within new products is a major productivity enhancer, and development staff should be encouraged to reuse code [Basili 91].

Schedule cycle time is calculated by adding the actual schedule duration times used for the implementation and qualification testing phases of the project. *Schedule adherence* is a measure of the organization's ability to develop products on time, and to meet commitments concerning schedules. As shown in the equation below, schedule adherence is calculated as the difference between the estimated schedule duration time and the actual schedule duration time. The measure is calculated as a percentage; a negative number indicates a schedule slip, while a positive number indicates that the actual development was done in less time than estimated.

Schedule adherence = % (Estimated schedule duration time - Actual schedule duration time) / Estimated schedule duration time.

Other common measures of organization performance which will not be considered here include:

- Development cost (which can be calculated as the effort multiplied by the average hourly labor rate for the project).

- Profitability.

- Customer satisfaction.

Examples of methods to calculate these measures can be found in [Möller 93].

4.6 Observing Trends in Performance

The impact of process improvement methods can be observed over time by observing the trend of the organization performance measures. The selection and definition of organization performance measures are dependent on the goals of the organization.

The performance measures defined here, however, correspond to goals that are common among many organizations.

4.6.1 Defect Trends

Many organizations wish to find defects early in their software development process and they want their customers to find as few defects as possible during field use. Methods such as formal inspections [Fagan 76] help organizations improve their ability to find defects early in their development process. An organization can observe the trend of the distribution of defects for various development phases over time. If the average cost is known for correcting a defect, then the savings can be calculated over time as more defects are found earlier in the development process. Similarly, the defect rates can be compared over time, and the resulting improvement in product quality can be observed. The defect rate for field use is often identified as a numerical target goal for organizations wishing to improve product quality within a specified time. A commonly used example target goal is Six Sigma performance which would equate to a field use defect rate of 0.0034 defects/KLOC. This is a very aggressive quality goal which most organizations today do not achieve. Capers Jones has identified quantitative targets for excellent software performance based upon data obtained from Baldrige Award winners [Jones 92]. He identifies a target performance value for product quality of less than 0.025 field use defects per function point per year. For products written in C, this would approximately correlate to a field use defect rate of less than 0.2 defects/KLOC.

4.6.2 Productivity Trends

Project productivity should improve as the software development process is improved. As product and process quality are improved, less rework will occur in the development process, and overall project productivity will increase. This improvement will be offset somewhat by the additional effort that will be required for improving the process, for example, the training time needed to learn a new method. Project productivity could also be affected negatively as a result of external influences, e.g., business climate changes, natural disasters (hurricanes).

4.6.3 Schedule Trends

Reducing schedule cycle time is a primary goal of most executives of organizations that produce products. This is based upon the observation that organizations that are first to market with a new product typically gain a larger share of the market. An organization may also wish to reduce the difference between its schedule estimates and actual performance. An organization with good repeatability of schedule prediction is in a stronger position to reduce schedule cycle times.

4.6.4 Process Improvement Goals

The goals and observation of trends resulting from process improvement are summarized below in Table 7. Many of the numerical targets are obtained from [Jones 92]. The numerical targets given may be difficult for many organizations to achieve depending on their current maturity level. By observation over time of the performance measures, one may conclude whether or not the software process improvement methods implemented are having an impact on an organization's performance.

Table 7: Summary of Process Improvement Goals

Business Goal	Performance Measure	Target Value	Target Trend
Find defects earlier	Defect detection distribution	Field use < 5%	Find >75% of defects through inspections
Improve product quality	Defect rate	<0.1 defects/ KLOC in field use	>40% per year decrease in defect rates
Increase productivity	Project productivity	Dependent on product size	>25% per year improvement
Reduce time to market	Schedule cycle time	Dependent on product size	reduce by >15% per year
Improve schedule predictability	Schedule adherence	no worse than -10%	Achieve target value and maintain within -10%-0%.

4.6.5 Return on Investment

Another observation that an organization may wish to make is the financial impact which resulted from implementation of the process improvement methods. This normally would be a return on investment (ROI) type of calculation in which the cost savings realized could be compared with the cost of investing in the process improvement methods. In [Rozum 93], a process improvement benefit index is defined as the amount saved as a result of process improvement divided by the cost of process improvement. The cost of process improvement would contain costs such as training personnel in a process improvement method, consultant fees, managing the improvement action activities, and purchasing tools. The amount saved could include reduced rework, less effort to produce a product, increased productivity, reduced maintenance costs, etc. To calculate your savings, you could use the change over time of the defect

detection distribution, if the average cost to correct a defect in each phase is known. This measure is somewhat difficult to apply since many of the costs and savings are not easily quantified or captured (such as improved customer satisfaction). We will attempt to calculate the process improvement benefit index in those case study site organizations where cost and savings data are available.

4.7 Performance Measures Conclusion

This report defines a basic set of measures to observe the performance of software development organizations over time. The details of definition, data collection, and use of the measures will be unique to each organization participating in the case study. The purpose of this report is to identify a limited number of simple measures that could be used for data collection and observation. The question to be studied is: What is the impact of software process improvement methods used within Siemens software development organizations? We anticipate that the performance of an organization will improve as its software development process is improved. The basic set of measures will help us to better understand the result of software process improvement. We will thus be better able to provide guidelines to other organizations wanting to improve their software development process. We recommend the defined performance measures to organizations that are planning to establish measurement and process improvement programs.

5 Baseline Performance Data

5.1 Data Collection

A metrics baseline was requested from each case study site. We asked each point-of-contact to fill out a data input form indicating the primary data, environmental data, and organization performance measures defined in Chapter 4 of this report. The data input form was accompanied by a white paper, similar in content to Chapter 4, discussing and defining the performance measures. A letter was sent to each point-of-contact indicating that the defined measures were merely examples, and that if better measures of performance were used by the organization, that data could also be reported. Since the goal of the project was to observe organization performance over time, the definitions of the measures could be unique to each case study site as long as they were consistently applied over time. It was suggested that the measurement baseline be calculated using a recent product release that had been used by customers in the field for one year.

5.2 Data Analysis

The following observations were made concerning the collection of the baseline performance data.

- All organizations considered the performance data as proprietary and confidential. Some organizations considered the data to be so sensitive that they refused to supply it for the study, although they participated in the interviews. The organizations stated that they needed to protect their data in order to control customer communications and interactions. We also suspect that they may have feared the use of the data for comparison across organizations. Although not explicitly stated, it is suspected that this fear may be greater within organizations that have multiple development sites within multiple countries. Organizations also expressed this fear of comparison to others (particularly across national barriers) when asked to perform self-assessments as part of a corporate thrust.

- Many of the organizations had difficulty collecting the performance data. The least amount of difficulty was exhibited in organizations that had established metrics programs. It was apparent that some of the organizations were collecting such data for the first time, and used the case study project as a motivator for measurement application. The other difficulty resulted from the fact that many of the organizations work on many product releases simultaneously, often with the same staff. For example, a staff member may be asked to maintain a

previously released software version, as well as be involved with design of future releases and implementation of the current release. Thus, it was often difficult to collect performance data so as to accurately separate effort, time, and defects among the various releases. Having staff members work on multiple releases was often used to meet customer needs, i.e., reduce the time increments between releases when schedule cycle times were long. In general, this was felt to be a complexity issue in managing the organization.

- Most of the organizations tended to adhere to the measurement definitions as given in Chapter 4. This may be a result of corporate training activities and standards developed over the past few years on measurement application.

- As anticipated, the environmental data were viewed as controversial and difficult to determine. For organizations that had not yet conducted an assessment, the maturity level estimates tended to differ significantly depending on who the point-of-contact asked within the organization. In general it was observed that newer staff tended to estimate the maturity level lower than staff who had worked within the organization for a longer time.

5.3 Initial Data

The initial baseline performance data are summarized here as a composite of all the organizations that reported data. Again, we anticipate that there are differences among the organizations concerning definition of measures, and the accuracy of the data collected. The composite data is provided merely to identify rough performance indicators and demonstrate the diversity among the case study sites. The reader should be cautioned not to draw strong conclusions from the data since not all organizations reported data for all measures. The primary data, environmental data, and organization performance measures collected from the case study sites are summarized in Tables 8 to 10 respectively. The first value given is a composite nominal value averaged over all the reported projects followed by a range of values collected.

Table 8: Primary Data - Summary

Measure	Definition	Units	Nominal Value	Range
Defects	Count of defects found during implementation, qualification testing, and field use phases	Number of defects found per phase	Implementation: 1400 Qualification testing: 650 Field use: 180	Implementation: 120-6300 Qualification testing: 130-2500 Field use: 1-450
Product Size	Lines of code (LOC) count excluding comment and blank lines	LOC	1.6M	150K-5M
Effort	Staff-hours required for product development through first year of field use	Staff-hours	300K	15K-750K
Schedule time duration - actual	Schedule time used per phase	Months per phase	Implementation: 14 Qualification testing: 7	Implementation: 4-20 Qualification testing: 2-16
Schedule time duration - estimated	Schedule time estimate per phase	Months per phase	Implementation: 13 Qualification testing: 5	Implementation: 4-22 Qualification testing: 2-13

Table 9: Environmental Data-Summary

Measure	Definition	Units	Nominal Value	Range
Staff Size	Current number of software engineers, testers, quality specialists, & documentation specialists	Number of people	450	25-1600
Staff turnover	% (No. staff members who have left in last 12 months)/Staff size	% people who left	14	4-21
Maturity level	CMM levels 1-5 from last assessment or estimate	1-5	2	1-4
Morale	Subjective estimate of staff morale	Poor, average, good, or excellent	Average	Average -good

Table 10: Organization Performance Measures - Summary

Measure	Definition	Units	Nominal Value	Range
Defect detection distrib- ution	% of defects found during implementation, qualification testing, and field use phases	% of defects found per phase	Implementation: 63 Qualification testing: 29 Field use: 8	Implementation: 23-68 Qualification testing: 27-78 Field use: 4-35
Defect rate	Defects divided by product size per phase	Defects/ KLOC per phase	Implementation: 0.8 Qualification testing: 0.6 Field use: 0.2	Implementation: 0.2-1.7 Qualification testing: 0.3-1.2 Field use: 0.008-0.36
Project product- ivity	Product size divided by effort	LOC/Staff -hour	5	2.6-6.7
Schedule cycle time	Sum of schedule duration times for implementation & qualification testing phases	Months	21	6-36
Schedule adherence	%(Estimated schedule duration time - Actual schedule duration time)/Estimated schedule duration time	%	-17	-42 – -13

6 Guidelines for Selecting Process Improvement Methods

Having observed the selection of software process improvement methods at the case study sites, we feel it is appropriate to provide some general guidance to organizations interested in process improvement. An approach to selecting and implementing software process improvement methods is summarized below and described in the following sections.

1. Establish improvement goals.

2. Identify improvement key process areas (KPAs).

3. Select process improvement methods.

4. Establish responsibility.

5. Communicate the process improvement plan.

6. Train.

7. Define progress tracking measures.

8. Implement the process improvement methods.

9. Collect and analyze tracking data.

10. Adjust the process improvement plan.

6.1 Establish Improvement Goals

The goals for software process improvement should be identified and communicated to the entire organization. The goals should be derived from and be consistent with the overall business goals of the organization. The goals of improved performance (e.g., improved quality, improved productivity, reduced cycle time, and better schedule adherence) should be described in accordance with the terminology of the specific corporate environment. Quantitative goals of what improvements are expected to be achieved in a specific time frame should be identified. The organization performance measures in Table 6 and the target trend values in Table 7 within Chapter 4 of this report can give some guidance for establishing the improvement goals. Some example improvement goals are given below:

- Reduce field defect rates by 50% within 3 years.

- Find at least 75% of all defects throu. h inspections within 2 years.

- Double project productivity within 4 years.

- Reduce schedule cycle time to 20 months within 5 years.

- Achieve implemented schedule to within 10% of the estimate within 3 years.

6.2 Identify Improvement Key Process Areas

The organization should identify the key process areas (KPAs) on which it will focus its process improvement efforts. These KPAs would typically be identified as findings during a software process assessment (SPA). It is important to select process improvement methods that are appropriate to the current maturity level of the organization, as identified in the Capability Maturity Model for Software (CMM) [Paulk 93]. The CMM identifies the following five levels of maturity of a development organization.

1. Initial. The development environment is unstable. The organization does not consistently apply software engineering management to the process, nor does it use modern tools and technology. Performance can only be predicted by individual, rather than organizational, capability. Level 1 organizations may have serious cost and schedule problems.

2. Repeatable. At Level 2, the organization has installed basic software management controls. Stable processes are in place for planning and tracking software projects. Project standards exist and are used.

3. Defined. At Level 3, the organization has a standard process for developing and maintaining software across the organization. The software engineering and software management processes are integrated into a coherent whole. A software engineering process group (SEPG) facilitates software process definition and improvement efforts. Organization-wide training is in place, to ensure that all employees have the skills necessary to perform their duties. Peer reviews are used to enhance product quality.

4. Managed. At Level 4, the organization sets quantitative quality goals for software products. Productivity and quality are measured for important software process activities across all projects in the organization. A process database is used to collect and analyze the data from a carefully designed process. There are well-defined and consistent measures for evaluating processes and products.

5. Optimizing. At Level 5, the organization is focused on continuous improvement. There are means of identifying weak processes and strengthening them. Statistical evidence is available on process effectiveness and is used in performing cost-benefit analyses on new technologies. Innovations that exploit the best software engineering practices are identified.

Some software process improvement methods and their corresponding primary key process areas of the CMM are given in Table 11.

Table 11: Software Process Improvement Methods

Method	Key Process Areas	CMM Level
Estimation	Software project planning	2
ISO 9000 certification	Software quality assurance Organization process def.	2 3
Software process assessment (SPA)	Organization process focus	3
Process definition	Organization process def.	3
Formal inspection	Peer reviews	3
Software measurement & metrics	Software project planning Software project tracking & oversight Integrated software mgt. Quantitative process mgt. Software quality mgt. Process change mgt.	2 2 3 4 4 5
Computer aided software engineering (CASE)	Software configuration mgt. Software quality assurance Software project tracking & oversight Organization process def. Software product engr.	2 2 2 3 3
Interdisciplinary group methods (IGMs)	Intergroup coordination	3
Software reliability engineering (SRE)	Quantitative process mgt.	4
Quality function deployment (QFD)	Software quality mgt.	4
Total quality management (TQM)	Organization process focus Quantitative process mgt. Software quality mgt. Process change mgt.	3 4 4 5
Defect prevention process (DPP)	Defect prevention	5
Cleanroom software development	Quantitative process mgt. Software quality mgt. Defect prevention	4 4 5

Key process areas of the CMM, and some example software process improvement methods are illustrated in Figure 6-1.
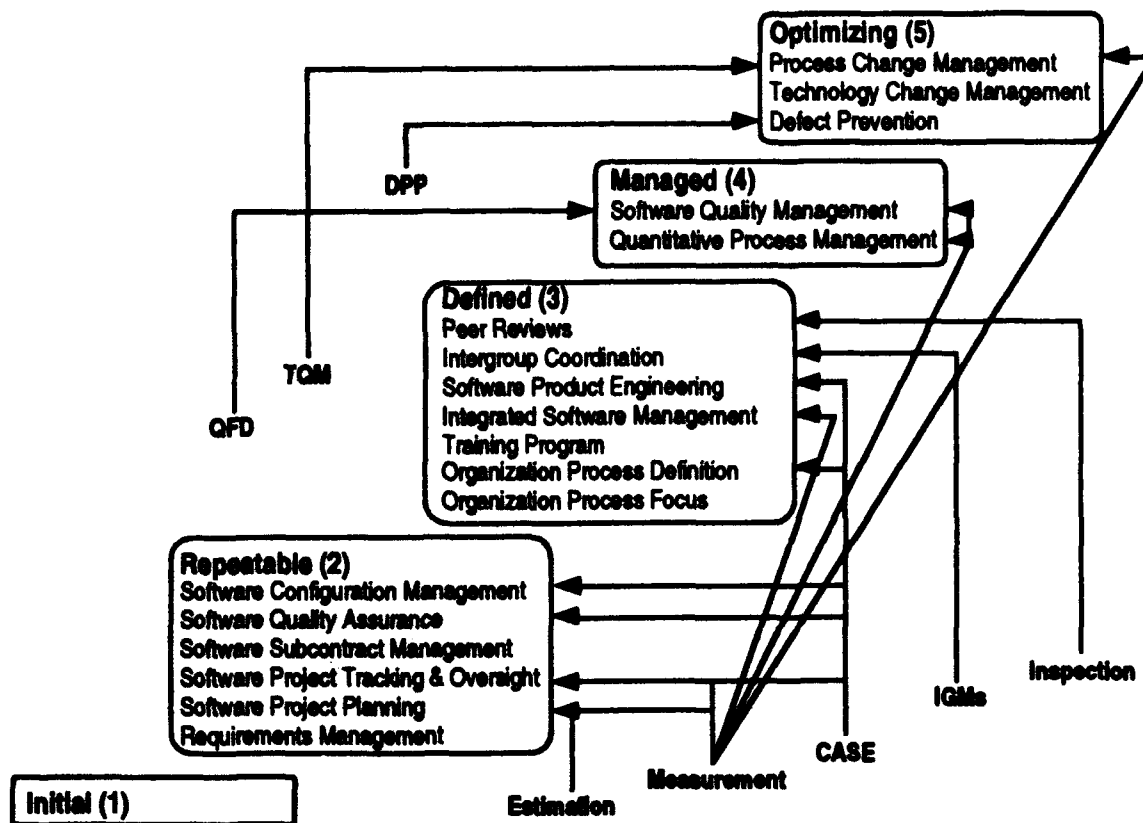
Figure 6-1: Key Process Areas

6.3 Select Process Improvement Methods

It is suggested that no more than three or four software process improvement methods be selected for achieving the KPAs of interest. It is recommended that you consider the following when selecting the methods.

- Appropriateness of method to maturity level and target KPAs.

- Consistency with your organization's goals.

- Potential benefit.

- Initial investment requirements.

- Implementation difficulty.

- Anticipated organizational resistance.

One tool which could assist you in selecting the methods is a matrix identifying selection criteria for each process improvement method. This matrix would list the methods being considered with a subjective score for key selection criteria. An example matrix is given in Table 12 where the criteria are assigned a value from 1-Low to 5-High.

6.4 Establish Responsibility

After an organization has chosen the appropriate process improvement methods, it should next establish responsibility for the software process improvement program. This is often assigned to a software engineering process group (SEPG) associated with a management steering committee. Specific improvement methods can be assigned for implementation to action teams responsible for introducing the method to the organization.

6.5 Communicate the Process Improvement Plan

The process improvement plan must be communicated to the entire organization so that its purpose is understood and accepted. This step is necessary to gain buy-in from the organization for the actions oriented towards software process improvement. The benefits of software process improvement must be communicated to everyone to obtain a high degree of cooperation and improvement.

6.6 Train

Training is necessary to transfer the software process improvement methods to the organization so they become accepted practices. Each method will most likely require specialized training for implementation. In addition, general training on software process improvement is desirable for all software engineering staff. In this manner, some of the environmental issues conducive to process improvement can be addressed, such as establishing a "quality culture."

Table 12: Example Matrix of Criteria for Selecting Process Improvement Methods

Method	Potential Benefit	Initial Investment	Implementation Difficulty	Anticipated Organizational Resistance
Estimation	2	1	1	1
ISO 9000	1	3	2	1
Software process assessment (SPA)	3	3	3	1
Process definition	3	3	4	2
Formal inspection	4	2	1	1
Measurement	4	2	4	4
Computer aided software engineering (CASE)	3	4	4	2
Interdisciplin. group methods	3	2	3	3
Software reliability engineering (SRE)	2	3	4	2
Quality function deployment (QFD)	3	2	3	2
Total quality management (TQM)	3	3	4	3
Defect prevention process (DPP)	5	4	4	3
Cleanroom	5	5	5	5

6.7 Define Progress Tracking Measures

Measures should be next defined for tracking the progress of the software process improvement methods. In some cases, the training given for the selected methods will provide suggested measures. Within this report, Table 6 of organization performance measures may prove useful. Suggestions for implementing a measurement process can also be found in [McAndrews 93].

6.8 Implement the Process Improvement Methods

The methods are then implemented so that they will eventually become part of the standard work practices of the organization. For the more complex methods, it is sug-

gested that a pilot project implementation be made prior to general introduction to the organization. As taught within the SEI Managing Technological Change Course, the time for acceptance and institutionalization of new methods may be excessive and barriers to acceptance of new ideas must be overcome. This is illustrated by the technology adoption curve (Figure 6-2) from the course taught by Deimel, Maher, and Myers.

6.9 Collect and Analyze Tracking Data

Once the measures are defined and the method implementation is initiated, the tracking data can be collected and analyzed. These data are important to observe the impact of the method and to establish how effective its implementation is.
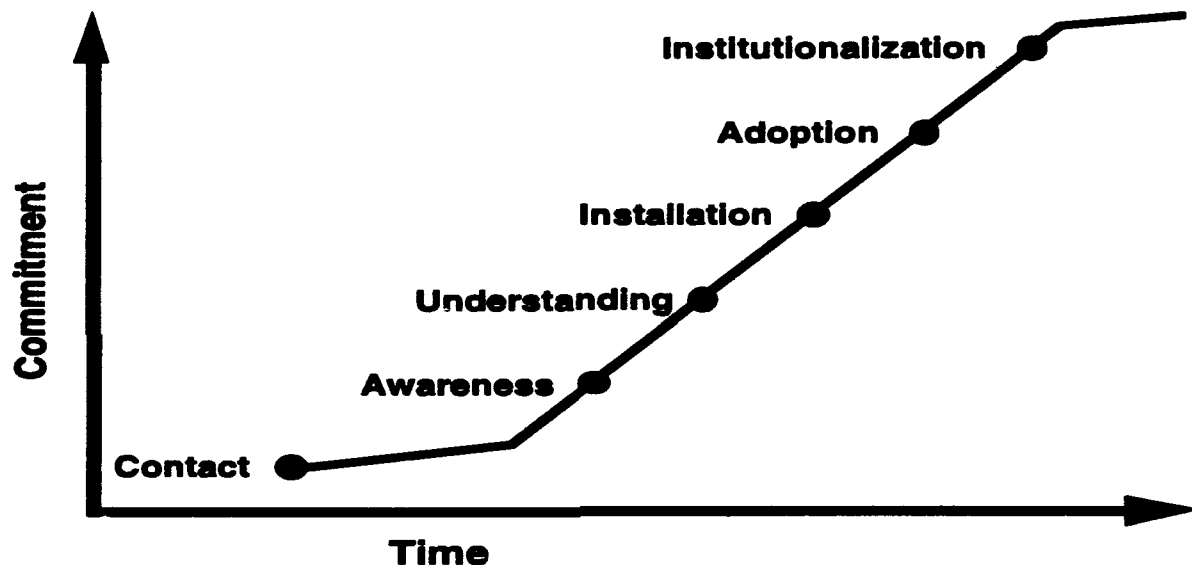
Figure 6-2. Technology Adoption Curve

6.10 Adjust the Process Improvement Plan

Analysis of the tracking data will provide insights into the effectiveness of the implementation of the software process improvement program and will help to identify what adjustments should be made to the plan. Possible alternatives include modifying the implementation by providing better or additional training and expanding the pilot project, for example. The data may suggest stopping or delaying implementation of certain software process improvement methods where acceptance barriers are large and difficult to overcome. The data will also suggest when the method has been successfully adopted. In this case, it is time to identify some new software process improvement methods for implementation. This software process improvement methods

selection approach can then be followed again from steps 1,2, or 3 as defined in Sections 6.1-6.3.

7 Common Implementation Problems and Introduction Hints

The observations given below identify some common barriers to implementation of software process improvement methods experienced by the case study sites, and some of the techniques used to overcome these barriers.

- *Getting started:* Some of the organizations had difficulty getting started with software process improvement and the methods they selected. We encouraged these organizations to undergo assessment as a proven technique for helping to identify their priorities and get buy-in across the organization for process improvement activities. We also encouraged them to establish a software engineering process group (SEPG). They were given guidance material and contact information for the SEI associated software process improvement network (SPIN) organizations, and more experienced Siemens development organizations.

- *Staff turnover:* Some of the organizations have been involved in downsizing (layoff) activities which affected software engineering staff turnover. This is a particularly difficult barrier for software process improvement since the improvement effort requires initial and sustained investment, and the impact of the improvement may not be measurable for a few years. We have also observed that within any organization, certain champions and advocates of software process improvement exist. If these individuals are affected or their priorities change as a result of downsizing, then introduction of new software process improvement methods is slower and more difficult. Organizations exhibiting downsizing were encouraged to consider profitability goals as part of the initiation and planning for software process improvement. Only one of the case study organizations had previously utilized profitability measures within the software development organization, although all the organizations were affected by overall business profitability goals. We have also observed that large staff turnover currently appears to be a greater problem at the U.S. case study sites than those in Germany.

- *Dedicated resources:* Some of the organizations utilized part-time resources, usually line managers or improvement teams, to implement software process improvement methods. Although this issue is greatly dependent on the size of the organization and the specific skills and influence of the individuals involved, part-time effort on process

improvement is usually not as effective as when full-time dedicated resources are used. Organizations using part-time resources for process improvement were encouraged to appreciate the return-on-investment of process improvement. They were also encouraged to introduce some of the methods on a pilot project basis until they were generally recognized as successful. The most effective observed SEPGs often had a full-time team leader supported by team members who were first-line managers dedicating approximately 20% of their time to process improvement. These managers had control of staff who could be applied to process improvement as required.

- *Management support:* We have generally observed that management support is necessary for software process improvement. To assist in overcoming this barrier, the point-of-contacts were offered support in communicating with and educating their management on the benefits of process improvement. The need for management support and buy-in at all levels of the organization seemed to be a more important issue within the organizations in the U.S. than for those in Germany. We also observed that organizations that provided quality training (e.g., TQM, Crosby), were generally more supportive of efforts oriented towards software process improvement.

- *Time restrictions:* Some organizations had difficulty finding the time to work on software process improvement since they had extreme commitments to deliver customer products. We pointed out in these cases that delivery dates had high priority, but that when the release was completed, that effort must be put into software process improvement in order to avoid future emergencies.

8   Conclusions and Recommendations

This report contains limited results because we have been observing the case study sites for only a short period of time. Several years are required in order to observe the impact of software process improvement. Nevertheless, some preliminary recommendations can be made at this time to organizations wishing to improve their software development process.

- Use the Capability Maturity Model (CMM) as a guide to suggest what actions should be taken to improve your software process. At Siemens case study sites, the CMM was relatively easy for organizations to understand and apply to their situation. It gives an organization a framework for determining which methods an organization should use to improve based upon its current maturity. Table 13 summarizes some of the implementation issues of some commonly applied software process improvement methods. For each method, we recommend a maturity level range at which an organization should be in order to consider implementing the method for the first time. In addition, based on our experience at Siemens, we provide a summary of our perceived pros and cons concerning implementation of the method.

- Use assessment to start a software process improvement program. Assessment is a very powerful method for identifying priorities for improvement and building consensus within the organization.

- Pick a few process improvement methods and implement them effectively. Many organizations make the mistake of initiating too many software process improvement activities which do not get implemented well.

- Pay attention to the implementation of the method as much as or more than the method itself. Sustained improvement over time is necessary, and thus, the selected methods must be implemented well. This includes the quality of training and management of the actions to be implemented.

Table 13: Implementation Issues Summary

Method	Consideration Levels	Pros	Cons
Estimation	1	Fundamental to project planning	Works best when historical data are available
ISO 9000	1	Required for many markets	Emphasis is on evidence rather than improvement
SPA	1-2	Good first step towards process improvement	Investment provides primarily findings
Process definition	1-2	Provides baseline for improvement	Representation tools skills often missing
Formal inspection	1-2	Easy to begin	More commonly used for code than documents
Measurement	1-2	Used with other methods	Must be tailored to an organization's specific goals
CASE	1-2	Automates process	High investment (e.g., license fees, training)
IGMs	1-2	Promotes better teamwork	Possible communication & meeting overhead
SRE	2-3	Provides field defect rate predictions	Training and skills often missing
QFD	2-3	Helps build the "right" products	Difficult to manage product complexity and communications
TQM	2-3	Builds a "quality culture"	Requires commitment & implementation throughout organization
DPP	3-4	Makes classes of errors extinct	Can only be considered by mature organizations
Cleanroom	3-4	Can result in high product quality	Different than current development practices

- Introduce improvement methods in an appropriate sequence. Some process improvement methods are easier to introduce and implement than others. For example, defect detection (formal inspection) must be implemented before defect prevention (DPP). Also, inspections are easier to introduce than a metrics program. Organizations at lower levels of the CMM should consider implementing the easier methods first so as to improve their return-on-investment and increase the likelihood of successful implementation. A common pattern observed

at a number of the case study sites is to begin implementing simple measures oriented towards the level 2 KPAs of software project planning and software project tracking and oversight. Once progress has been made on achieving the level 2 KPAs, formal inspections are implemented. As inspections are implemented, further measurement data become available and are used for better understanding of the development process with respect to measures such as where defects are found. As the level 3 and 4 KPAs are implemented, the organization starts looking at defect prevention process (DPP) in order to reduce the number of defects introduced to the process.

- Consider cultural factors when implementing process improvement. Country cultural factors are substantial. Although the case study sites are quite diverse organizations with differing products and development processes, the diversity appears to be greatest between organizations residing in Germany and those residing in the U.S. In many cases, the same software process improvement methods were selected and implemented at case study sites in Germany and the U.S., often using the same training courses and trainers. However, the way that the methods were introduced and the level of acceptance of the methods, were very different between the German and U.S. sites. This implies that the cultural characteristics of an organization have a significant impact on its success with adopting software process improvement methods. It is recommended that software engineering managers and other staff members involved with organizational change (e.g., SEPG members) pay particular attention to the cultural characteristics of the organization when implementing selected process improvement methods.

9 References

[AMI 92]   *A Quantitative Approach to Software Management*, ESPRIT
              AMI Project Handbook. 1992.

[Basili 91]   Basili, V. R., and Rombach, H. D. "Support for Comprehensive
              Reuse." *Software Engineering Journal* (Sept. 1991): 303-316.

[Card 93]   Card, D., ed. "Bootstrap: Europe's Assessment Method."
              IEEE Software (May 1993): 93-95.

[Carleton 92] Carleton, A. D. *Software Measurement for DoD Systems:
              Recommendations for Initial Implementation* (CMU/SEI-92-
              TR-19, ADA 258305). Pittsburgh, Pa.: Software Engineering
              Institute, Carnegie Mellon University, September 1992.

[Fagan 76]   Fagan, M. E. "Design and Code Inspections to Reduce Errors
              in Program Development." *IBM Systems Journal*, Vol. 15, No.
              3: 182-210.

[Florac 92]   Florac, W. A. *Software Quality Measurement: A Framework
              for Counting Problems, Failures and Faults* (CMU/SEI-92-TR-
              22, ADA 258556). Pittsburgh, Pa.: Software Engineering
              Institute, Carnegie Mellon University, September 1992.

[Goethert 92] Goethert, W. B.; Bailey, E. K.; Busby, M. B. *Software Effort
              Measurement: A Framework for Counting Staff-Hours*
              (CMU/SEI-92-TR-21, ADA 258279). Pittsburgh, Pa.: Software
              Engineering Institute, Carnegie Mellon University, September
              1992.

[Humphrey 89] Humphrey, W. S. *Managing the Software Process*. Reading,
              Mass.: Addison-Wesley Publishing Company, 1989.

[IFPUG 90]   *Function Point Counting Practices Manual*. International
              Function Point Users Group.

[Jilek 92]   Jilek, P.; Möller, K.-H.; Paulish, D. J. "The Use of Metrics for
                  Software Development." *Proceedings of the Ninth World
                  Conference on Computer Security, Audit, and Control.*
                  Elsevier Advanced Technology, November 1992.

[Jones 92]   Jones, C. "Being Best in Class." Software Productivity
                  Research Report, December 1992.

[Kellner 92]   Kellner, M. I., and Over, J. W. "A Software Quality
                  Improvement Framework," *Proceedings of the Software
                  Engineering Forum.* Milan, Italy; June, 1992.

[McAndrews 93] McAndrews, D. R. *Establishing a Software Measurement Pro-
                  cess* (CMU/SEI-93-TR-16, ADA 267896). Pittsburgh, Pa.:
                  Software Engineering Institute, Carnegie Mellon University,
                  July 1993.

[Möller 93]   Möller, K.-H., and Paulish, D. J. *Software Metrics: A
                  Practitioner's Guide to Improved Product Development.* Los
                  Alamitos, Calif.: IEEE Computer Society Press, 1993.

[Park 92]   Park, R. E. *Software Size Measurement: A Framework for
                  Counting Source Statements* (CMU/SEI-92-TR-20, ADA
                  258304). Pittsburgh, Pa.: Software Engineering Institute,
                  Carnegie Mellon University, September 1992.

[Paulk 93]   Paulk, M. C. et al. *Capability Maturity Model for Software,
                  Version 1.1* (CMU/SEI-93-TR-24, ADA 263403). Pittsburgh,
                  Pa.: Software Engineering Institute, Carnegie Mellon
                  University, February 1993.

[Rozum 93]   Rozum, J. A. *Concepts on Measuring the Benefits of Software
                  Process Improvement* (CMU/SEI-93-TR-9, ADA 266994).
                  Pittsburgh, Pa.: Software Engineering Institute, Carnegie
                  Mellon University, June 1993.

[Siemens 89]   Siemens Norm for Lines of Code. *Software Definition der
                  zeilenbezogenen Programmgrosse Masseinheit LOC (Lines
                  of Code)* (SN77353). May 1989.

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified		1b. RESTRICTIVE MARKINGS None			
2a. SECURITY CLASSIFICATION AUTHORITY N/A		3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release Distribution Unlimited			
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A					
4. PERFORMING ORGANIZATION REPORT NUMBER(S) CMU/SEI-93-TR-26		5. MONITORING ORGANIZATION REPORT NUMBER(S) ESC-TR-93-200			
6a. NAME OF PERFORMING ORGANIZATION Software Engineering Institute	6b. OFFICE SYMBOL (if applicable) SEI	7a. NAME OF MONITORING ORGANIZATION SEI Joint Program Office			
6c. ADDRESS (city, state, and zip code) Carnegie Mellon University Pittsburgh PA 15213		7b. ADDRESS (city, state, and zip code) HQ ESC/ENS 5 Eglin Street Hanscom AFB, MA 01731-2116			
8a. NAME OF FUNDING/SPONSORING ORGANIZATION SEI Joint Program Office	8b. OFFICE SYMBOL (if applicable) ESC/ENS	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F1962890C0003			
8c. ADDRESS (city, state, and zip code)) Carnegie Mellon University Pittsburgh PA 15213		10. SOURCE OF FUNDING NOS.			
		PROGRAM ELEMENT NO 63756E	PROJECT NO. N/A	TASK NO N/A	WORK UNIT NO. N/A

11. TITLE (Include Security Classification)
Case Studies of Software Process Improvement Methods

12. PERSONAL AUTHOR(S)
Daniel J. Paulish, Resident Affiliate, Siemens Corporate Research, Inc.

13a. TYPE OF REPORT Final	13b. TIME COVERED FROM TO	14. DATE OF REPORT (year, month, day) December 1993	15. PAGE COUNT 54 pp.

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB. GR.	case studies Siemens
			measurement
			process improvement methods

19. ABSTRACT (continue on reverse if necessary and identify by block number)

This report describes the case studies approach applied at a number of Siemens software development organizations to observe the impact of software process improvement methods. In addition, the report provides guidance to software development organizations that want to improve their processes. A set of organization performance measures are defined to help an organization observe its software process improvement over time. An approach is given for selecting software process improvement methods. The report concludes with a description of common implementation problems, and recommendations for organizations to improve their software processes.

(please turn over)

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ■ SAME AS RPT □ DTIC USERS ■	21. ABSTRACT SECURITY CLASSIFICATION Unclassified, Unlimited Distribution	
22a. NAME OF RESPONSIBLE INDIVIDUAL Thomas R. Miller, Lt Col, USAF	22b. TELEPHONE NUMBER (include area code) (412) 268-7631	22c. OFFICE SYMBOL ESC/ENS (SEI)

ABSTRACT — continued from page one, block 19